Assignment 1

Chat Server with Groups and Private Messages

Course: CS425: Computer Networks

Instructor: Adithya Vadapalli

TAs Incharge: Rohit and Naman

Submission Deadline: 30.01.2025

Objective

Develop a multi-threaded chat server that supports private messages, group communication, and user authentication. This assignment will help you understand socket programming, multithreading, and data synchronization in networked systems.

Instructions

1. Clone the repository from: `https://github.com/privacy-iitk/cs425-2025.git`
2. Go to the `Homeworks/A1` directory as `cd cs425-2025/Homeworks/A1`
3. You will see the client code, a users.txt file, and a `Makefile` there.
4. This assignment can be solved in a group of at most three students.
5. Use only Piazza to ask for help in the assignment.
6. Your goal is to write the server code so that `make` compiles both the server and client codes (Note that the Makefile already has the compilation command for the server code!)

Requirements

1. Basic Server Functionality

1. Implement a TCP-based server that listens on a specific port (e.g., `12345`).
2. The server should accept multiple concurrent client connections.
3. Maintain a list of connected clients with their usernames.

2. User Authentication

1. Create a `users.txt` file to store usernames and passwords in the format `username:password`.
2. On client connection, prompt the user to enter their username and password.
3. Disconnect clients that fail authentication.

3. Messaging Features

We should allow clients to send messages to:

1. **All users:** Broadcast messages to all connected clients using `/broadcast <message>`
2. **Specific users:** Send private messages to a specific user using `/msg <username> <message>`
3. **Groups:** Send messages to all group members using `/group_msg <group_name> <message>`

4. Group Management

You should implement the following group commands:

1. `/create_group <group_name>`: Create a new group.
2. `/join_group <group_name>`: Join an existing group.
3. `/leave_group <group_name>`: Leave a group.

Furthermore, you should maintain a mapping of group names to their members.

1. Any client should be able to create a group.
2. The server will be maintaining the groups.

5. Commands

In summary the following commands have to be supported.

1. `/msg <username> <message>`: Send a private message to a user.
2. `/broadcast <message>`: Send a message to all users.
3. `/join_group <group_name>`: A user joins a group that has been created.
4. `/group_msg <group_name> <message>`: Send a message to a group.
5. `/leave_group <group_name>`: A user leaves a group.

Hints

1. If there is a shared resource accessed by multiple threads, we use `std::lock_guard <std::mutex>` to ensure thread-safe access. This prevents data races when multiple threads (handling different clients) try to read or modify the shared resource simultaneously. The lock guarantees that only one thread can access clients at a time, ensuring consistency and avoiding undefined behavior.

2. Server can maintain the list of users (everyone who is allowed to login), the clients (users who have logged in) and the groups created as follows.

```cpp
 std::unordered_map<int, std::string> clients; // Client socket -> username
std::unordered_map<std::string, std::string> users; // Username -> password
std::unordered_map<std::string, std::unordered_set<int>> groups; // Group -> client sockets
```

3. We have given you some example code to see how multithreading can be done. You can access it as: `cd cs425-2025/classroom-code/Threading`.

4. To implement the functionalities, the server modifies these maps. For instance, when a client joins a group, the server adds that client socket to the unordered set corresponding to the group name.

5. You should think about commands that the clients will run as messages to the server. For instance, when a client creates a group cs425 by running the command `/create_group cs425`. This should be equivalent to sending the string `/create_group cs425` to create a group at this end. Similarly, for other commands like `/join cs425`, the string "`/join cs425`" is sent to the server by the client. Then, the server parses the string and executes the actions that the string corresponds to.

6. Parsing a string from the server side can be done as follows:

```
if (message.starts_with("/group_msg")) {
    size_t space1 = message.find(' ');
    size_t space2 = message.find(' ', space1 + 1);
    if (space1 != std::string::npos && space2 != std::
        string::npos) {
        std::string group_name = message.substr(space1 +
            1, space2 - space1 - 1);
        std::string group_msg = message.substr(space2 + 1)
            ;
        group_message(client_socket, group_name, group_msg
            );
    }
}
```

Deliverables

1. Source code for the chat server.
2. `users.txt` file with test usernames and passwords.
3. A `README.md` file with instructions to compile and run the server. Your `README.md` should be really elaborate in describing your code. Therefore, you should also document not just how to run the code but also how your code works!
4. Put all the above things, including the client code, in a zip file.
5. Submission instructions will be given a week.

Expected Code Output

In one of the terminals, you should run the server. Each of the clients would run different terminals. Some possible outputs could be like the below.

```
$ .\server_grp
```

Listing 1: Running the server executable in one Terminal

```
$ .\client_grp
Connected to the server.
Enter username: alice
Enter password: password123
```

```
Welcome to the chat server!
bob has joined the chat.
frank has joined the chat.
/msg bob I will start a group CS425
[bob]:  alice great, I will join
/create_group CS425
Group CS425 created.
/group_msg CS425 Hi, Welcome to CS425
[Group CS425]: I have joined too
[Group CS425]: Started A1?
```

Listing 2: Running one of the clients in the second Terminal

```
$ .\client_grp
Connected to the server.
Enter username: bob
Enter password: qwerty456
Welcome to the chat server!
frank has joined the chat.
[alice]: bob I will start a group CS425
/msg alice great, I will join
/join_group CS425
You joined the group CS425.
[Group CS425]: Hi, Welcome to CS425
[Group CS425]: I have joined too
/group_msg CS425 Started A1?
```

Listing 3: Running another client in the third Terminal

```
$ .\client_grp
Connected to the server.
Enter username: frank
Enter password: letmein
Welcome to the chat server!
/join_group CS425
You joined the group CS425.
/join_group CS425
You joined the group CS425.
/group_msg CS425 I have joined too
[Group CS425]: Started A1?
```

Listing 4: Running another client in the fourth Terminal; Note that frank does not receive the group message till he joins the group

```
$ .\client_grp
Connected to the server.
Enter username: david
Enter password: qwerty
Authentication failed.
```

Listing 5: Running another client in the fifth Terminal

Grading Rubric

- **Correctness (60%)**: The server works as expected and supports all required features.
- **Code Quality (15%)**: Clean, modular, and well-documented code.
- **Documentation (25%)**: Clear instructions and explanation in the README.

Note: For any clarifications, contact the TA in charge. Ensure all submissions are made before the deadline. Late submissions will incur a penalty as per the course policy.

Instructor: **Adithya Vadapalli** TA Incharge: **Rohit and Naman**